



# Curso de Expressões Regulares

Autor: Leonardo Leitão

# Índice

1. Introdução .....	2
1.1. Visão Geral do Curso .....	2
1.2. Assine Nossa Canal .....	2
1.3. Repositório do Curso .....	2
1.4. Configuração do Ambiente .....	3
1.5. Autores .....	4
1.6. Suporte .....	4
2. Executando Expressões Regulares .....	5
2.1. Entendendo as Flags .....	5
2.2. Executando Regex em JS .....	5
2.3. Executando Regex em Ruby .....	5
2.4. Executando Regex em Python .....	6
2.5. Executando Regex em Go .....	6
2.6. Executando Regex em Java .....	7
3. Caracteres .....	9
3.1. Olá Regex! .....	9
3.2. Alguns Cuidados .....	9
3.3. Caracteres Simples .....	9
3.4. Meta-Caracteres .....	10
3.5. Meta-Caracteres: Ponto .....	11
3.6. Desafio: Lista de Arquivos .....	11
3.7. Selecionando Caracteres Brancos .....	12
3.8. Desafio: Três Espaços .....	12
3.9. Meta-Caracteres: Pipe (Ou) .....	12
3.10. Entendendo o Problema com Ponto .....	13
3.11. Selecionando Caracteres Unicode .....	13
4. Conjuntos .....	14
4.1. Trabalhando com Conjuntos .....	14
4.2. Trabalhando com Intervalos .....	14
4.3. Conjuntos e Meta-Caracteres .....	14
4.4. Alguns Cuidados com Intervalos .....	15
4.5. Usando Shorthands .....	15
4.6. Conjuntos Negados .....	16
4.7. Selecionando Intervalos Unicode .....	16
5. Quantificadores .....	17
5.1. Conhecendo os Quantificadores .....	17
5.2. Quantificador: ? (Zero-Um) .....	17
5.3. Quantificador: + (Um-Mais) .....	18

5.4. Quantificador: * (Zero-Mais) .....	18
5.5. Quantificador: {n, m} .....	18
5.6. Gulos vs Não Gulos .....	19
5.7. Apresentando Desafios .....	19
5.8. Desafio: Selecionando CPF .....	20
5.9. Desafio: Selecionando Telefone .....	20
5.10. Desafio: Selecionando E-mail .....	21
6. Grupos .....	22
6.1. Usando Grupos .....	22
6.2. Grupos & Retrovisores .....	22
6.3. Retrovisores: Só por Curiosidade .....	22
6.4. Grupos Aninhados .....	23
6.5. Alguns Cuidados com Grupos .....	23
6.6. Grupos Especiais #01 .....	24
6.7. Grupos Especiais #02 .....	24
7. Bordas .....	25
7.1. Usando Bordas .....	25
7.2. Implementando Dotall em JS .....	25
7.3. Usando a Flag Multiline .....	25
7.4. Bordas de Palavras .....	26
8. Receitas (Exercícios) .....	27
8.1. Aplicando Syntax Highlight #01 .....	27
8.2. Aplicando Syntax Highlight #02 .....	28
8.3. Selecionando Telefones .....	29
8.4. Selecionando Intervalos Numéricos .....	30
8.5. Selecionando Endereços IPv4 .....	30
8.6. Validação de Senha .....	31
8.7. Selecionando E-mail .....	32
9. Conclusão .....	33
9.1. Obrigado e Até Breve .....	33
Appendix A: Tabela de Códigos .....	34
Glossário .....	36

## **Sumário**

*Apostila do curso de Expressões Regulares da Cod3r.*

*Mais informações em <https://www.cod3r.com.br>*

# 1. Introdução

## 1.1. Visão Geral do Curso

O curso está dividido nas seguintes seções:



Figura 1. Visão Geral do Curso

## 1.2. Assine Nossa Canal

Estamos também no **Youtube!** Temos um canal no youtube com várias séries de vídeos para complementar o seu aprendizado, então gostaria de te convidar para fazer parte da nossa comunidade em <https://www.youtube.com/aulasdeprogramacao>

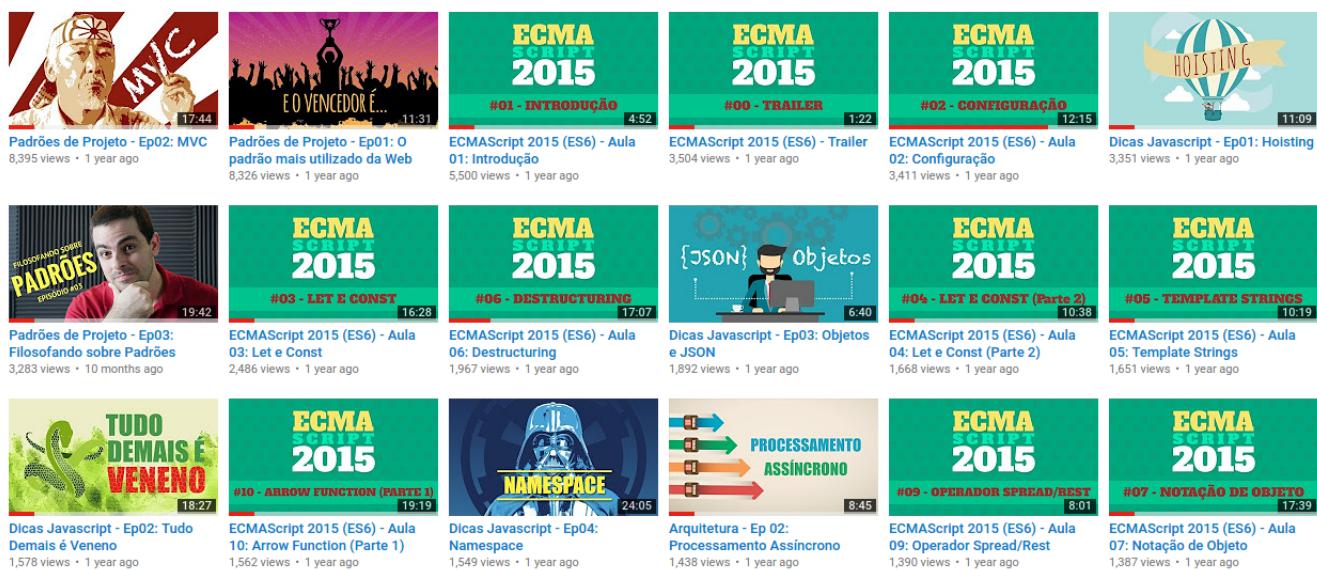


Figura 2. Algumas Aulas do Canal

## 1.3. Repositório do Curso

Precisa ter acesso ao **código fonte** dos exercícios do curso? Todos os exemplos desenvolvidos

durante as aulas estão disponíveis no **Github** da **Cod3r**: <https://github.com/cod3rcursos/curso-regex>



Se você não nunca trabalhou com repositórios Git, uma excelente ferramenta para facilitar a obtenção do código pode ser obtida no seguinte endereço:

<https://desktop.github.com/>

É possível ter acesso ao código fonte do curso sem precisar instalar nenhuma ferramenta, baixando diretamente o arquivo **.ZIP** da página do **Github**: <https://github.com/cod3rcursos/curso-regex/archive/master.zip>

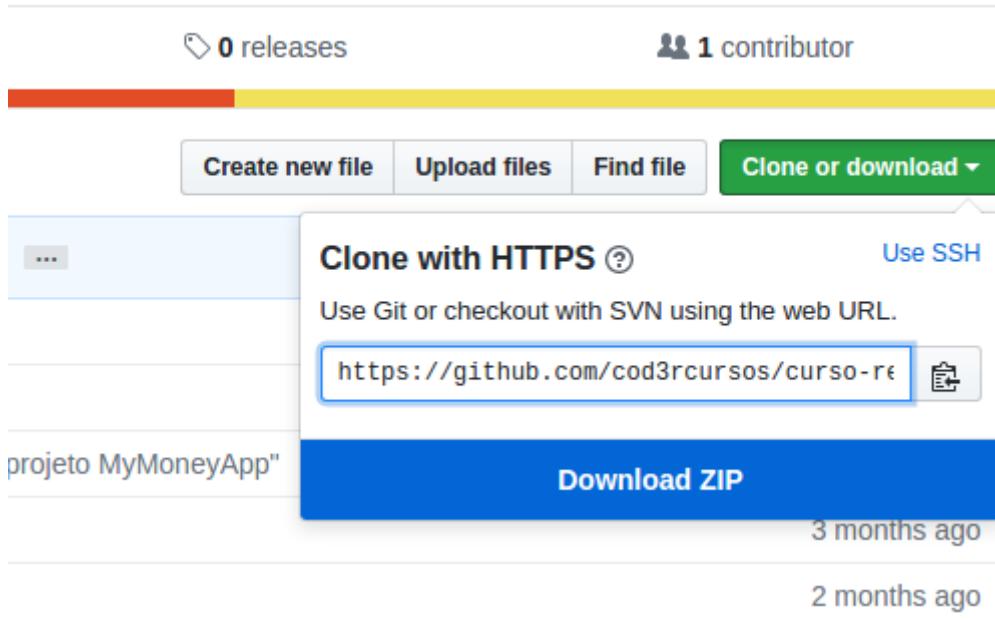


Figura 3. Opção para Baixar Repositório como .ZIP

## 1.4. Configuração do Ambiente

Inicialmente, será necessária a instalação de dois softwares:

1. Editor de texto de sua preferência (**Usaremos o VSCode**)
2. Node na versão 8.1 ou superior



Você pode obter a última versão do **VSCode** em  
<https://code.visualstudio.com/Download>



Você pode obter a última versão do **Node** em <https://nodejs.org/en/download/>



Caso seja necessária a instalação de outra ferramenta específica para a execução de algum exercício, será indicado na seção apropriada.

## 1.5. Autores



**Leonardo Leitão** é graduado em Engenharia Elétrica pela Universidade Federal do Ceará e Mestre em Informática Aplicada pela Universidade de Fortaleza, na qual trabalhou com Integração de Redes de Sensores sem Fio e Computação em Nuvem. Há doze anos atua na área como desenvolvedor de softwares e atualmente trabalha na DATAPREV como arquiteto, desenvolvendo sistemas para o INSS. Professor de desenvolvimento de software há dez anos.

Em 2016 foi um dos fundadores da **Cod3r**, na qual atua na criação de cursos online voltados para desenvolvimento de software. Com mais de 14.000 alunos, a Cod3r tem se tornado uma referência em ensino de qualidade no Brasil na área de programação.

## 1.6. Suporte

Não basta um curso ter qualidade, é fundamental um bom suporte para que você tenha uma excelente experiência no aprendizado. E nossa equipe estará à disposição para sanar eventuais dúvidas que você tenha durante as aulas.

E pra que a gente consiga oferecer um melhor suporte e de forma mais rápida, é importante que algumas regras sejam seguidas:

- **Manter perguntas separadas** - Não aproveite perguntas de outros alunos para fazer novas perguntas. Mesmo quando o erro parece ser o mesmo, as causas geralmente são diferentes.
- **Disponibilize seu código no Git** - Quando um problema ocorre, é muito comum eu ter que analisar o seu código para saber de fato o que está gerando o problema reportado, então disponibilize o seu código em algum repositório público. Poder ser o Github, Bitbucket, Gitlab... Você escolhe!
- **Suporte dentro do escopo** - Manter as dúvidas sobre questões relacionadas à execução dos exercícios e as dúvidas conceituais relativas ao escopo do curso. É inviável para nós professores, analisar problemas específicos ou implementar soluções para atender necessidades específicas de um aluno.
- **Curso está aberto** - Se entendermos que existe a demanda de adicionar um tópico que seja necessidade de um conjunto expressivo de alunos e que julgarmos que faz parte do escopo do curso, pode ter certeza que iremos acrescentar o material com o maior prazer.

Conto com vocês e pode ter certeza que estaremos presente na plataforma para te ajudar!

## 2. Executando Expressões Regulares

### 2.1. Entendendo as Flags

#### Listagem 1 - Flags

executandoRegex/Flags.js

```
// g - global  
// i - ignore case  
  
const texto = 'Carlos assinou o abaixo-assinado.'  
console.log(texto.match(/C|ab/))  
console.log(texto.match(/c|ab/i))  
console.log(texto.match(/ab|c/gi))
```

### 2.2. Executando Regex em JS

#### Listagem 2 - Usando JS

executandoRegex/UsandoJs.js

```
const texto = '0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f'  
  
const regexNove = RegExp('9')  
console.log('Métodos da RegExp...')  
console.log(regexNove.test(texto))  
console.log(regexNove.exec(texto))  
  
const regexLetras = /[a-f]/g  
console.log('Métodos da string...')  
console.log(texto.match(regexLetras))  
console.log(texto.search(regexLetras))  
console.log(texto.replace(regexLetras, 'Achei'))  
console.log(texto.split(regexLetras))
```

### 2.3. Executando Regex em Ruby

### Listagem 3 - Usando Ruby

executandoRegex/UsandoRuby.rb

```
texto = '0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f'
regexNove = Regexp::new('9')
puts regexNove.match(texto)

regexNove = %r{9}
puts regexNove.match(texto)

p regexNove =~ '894'

regexLetras = /[a-f]/
puts texto.scan(regexLetras).join(' ')

puts texto.split(/,/).join

print texto.split(/[aeiou]/)
```

## 2.4. Executando Regex em Python

### Listagem 4 - Usando Python

executandoRegex/UsandoPython.py

```
import re

texto = '0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f'

pattern9 = re.compile('9')
match1 = re.search(pattern9, texto)
print "Posicoes: %s, %s; Valor: %s." % (match1.start(), match1.end(),
match1.group(0))

valores = re.findall('[a-f]', texto)
print "Valores: %s" % valores
```

## 2.5. Executando Regex em Go

## Listagem 5 - Usando Go

executandoRegex/UsandoGo.go

```
package main

import (
    "bytes"
    "fmt"
    "regexp"
)

func main() {
    texto := "0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f"

    regex9, _ := regexp.Compile("9")
    fmt.Println(regex9.MatchString(texto))
    fmt.Println(regex9.FindString(texto))
    fmt.Println(regex9.FindStringIndex(texto))

    regexLetras, _ := regexp.Compile("[a-f]")
    fmt.Println(regexLetras.FindAllString(texto, 20))
    fmt.Println(regexLetras.ReplaceAllString(texto, "Achei"))

    resultado := regexLetras.ReplaceAllFunc([]byte(texto), bytes.ToUpper)
    fmt.Println(string(resultado))
}
```

## 2.6. Executando Regex em Java

## Listagem 6 - Usando Java

executandoRegex/UsandoJava.java

```
import java.util.regex.*;  
  
public class UsandoJava {  
    public static void main(String[] args) {  
        String texto = "0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f";  
        Pattern patternLetras = Pattern.compile("[a-f]");  
        Matcher matcher = patternLetras.matcher(texto);  
  
        while (matcher.find()) {  
            System.out.printf("Posicoes: %s, %s\tValor: %s%n",  
                matcher.start(), matcher.end(), matcher.group());  
        }  
  
        System.out.println(texto.replaceAll("[7-9]", "Achei"));  
    }  
}
```

## 3. Caracteres

### 3.1. Olá Regex!

#### Listagem 7 - Olá Regex

caracteres/OlaRegex.js

```
const texto = 'Casa bonita é casa amarela da esquina com a Rua ACASALAR.'  
  
const regex = /casa/gi  
console.log(texto.match(regex))  
console.log(texto.match(/a b/))
```

### 3.2. Alguns Cuidados

#### Listagem 8 - Alguns Cuidados

caracteres/AlgunsCuidados.js

```
const textoUmaLinha = '...' // aspas simples ou duplas  
  
const textoMultiLinha = `  
    linha 1  
    linha 2  
`  
  
// cuidado com o tab!  
console.log('    '.match(/\s/g))
```

### 3.3. Caracteres Simples

## Listagem 9 - Caractere Simples

caracteres/CaractereSimples.js

```
const texto = '1,2,3,4,5,6,a.b c!d?e'

const regexVirgula = ,/
console.log(texto.split(regexVirgula))
console.log(texto.match(regexVirgula))

console.log(texto.match(,/g))
console.log(texto.match(/A/g))
console.log(texto.match(/A/gi))
console.log(texto.match(/2/g))
console.log(texto.match(/b c!d/))
```

## 3.4. Meta-Caracteres

### Metacaracteres - Representantes

Metacaractere	Nome	Significado
.	Ponto	Um caractere qualquer
[]	Conjunto	Conjunto de caracteres permitidos
[^]	Conjunto Negado	Conjunto de caracteres proibidos

Figura 4. Meta-caracteres Representantes

### Metacaracteres - Quantificadores

Metacaractere	Nome	Significado
?	Opcional	Zero ou um
*	Asterisco	Zero ou mais
+	Mais	Um ou mais
{n, m}	Chaves	De n até m

Figura 5. Meta-caracteres Quantificadores

### Metacaracteres - Âncoras

Metacaractere	Nome	Significado
^	Circunflexo	Início de linha
\$	Cifrão	Fim de linha
\b	Borda	Início ou fim de palavra

Figura 6. Meta-caracteres Âncoras

## Outros Metacaracteres

Metacaractere	Nome	Significado
\	Escape	Uso de metacaracteres como literal
	Ou	Operação de Ou
( )	Grupo	Define um Grupo
\1...\9	Retrovisor	Regasta grupos já definidos

Figura 7. Outros Meta-caracteres

### Listagem 10 - Meta-caracteres

caracteres/Metacaracteres.js

```
// . ? * + - ^ $ | [ ] { } ( ) \ :  
const texto = '1,2,3,4,5,6,a.b c!d?e'  
const regexPonto = /\.g  
console.log(texto.split(regexPonto))  
  
const regexSimbolos = /,|\.|\?|!| /g  
console.log(texto.split(regexSimbolos))
```

## 3.5. Meta-Caracteres: Ponto

### Listagem 11 - Meta-caractere Ponto

caracteres/Ponto.js

```
// . é um coringa, válido para uma posição  
  
const texto = '1,2,3,4,5,6,7,8,9,0'  
  
console.log(texto.match(/1.2/g))  
console.log(texto.match(/1..2/g))  
console.log(texto.match(/1.../g))  
  
const notas = '8.3,7.1,8.8,10.0'  
console.log(notas.match(/8../g))  
console.log(notas.match(/.\../g))
```

## 3.6. Desafio: Lista de Arquivos

## Listagem 12 - Desafio Lista de Arquivos

caracteres/DesafioListaArquivos.js

```
const texto = 'lista de arquivos mp3: jazz.mp3,rock.mp3,podcast.mp3,blues.mp3'  
console.log(texto.match(/\.\mp3/g))  
  
// no futuro...  
console.log(texto.match(/\w+\.\mp3/g))
```

## 3.7. Selecionando Caracteres Brancos

### Listagem 13 - Caracteres Brancos

caracteres/Brancos.js

```
const texto = `  
ca r  
r o s!  
`  
  
console.log(texto.match(/ca\tr\nr\to\ss!/))
```

## 3.8. Desafio: Três Espaços

### Listagem 14 - Desafio dos três espaços

caracteres/DesafioTresEspacos.js

```
const texto = 'a b'  
console.log(texto.match(/a b/))  
console.log(texto.match(/a\s\s\sb/))  
  
// no futuro...  
console.log(texto.match(/a\s+b/))  
console.log(texto.match(/a {3}b/))  
console.log(texto.match(/a\s{3}b/))
```

## 3.9. Meta-Caracteres: Pipe (Ou)

## Listagem 15 - Pipe (Ou)

caracteres/Ou.js

```
const texto = 'Você precisa responder sim, não ou não sei!'
console.log(texto.match(/sim|não|sei/g)) // alternativa (OU)
```

## 3.10. Entendendo o Problema com Ponto

### Listagem 16 - Problema do Ponto

caracteres/ProblemaPonto.js

```
const texto = 'Bom\ndia'
console.log(texto.match(/.../gi))
console.log(texto.match(/.../gi)) // o ponto não engloba o \n

// dotall - alguns linguagens tem um flag /exp/s, mas JS não!
```

## 3.11. Selecionando Caracteres Unicode

### Listagem 17 - Caracteres Unicode

caracteres/Unicode.js

```
// no início...
// Um byte (8 bits) - 256 caracteres
// Símbolos, Pontuação, A-Z, a-z, 0-9

// Dois bytes (16 bits) - 65500+ caracteres
// +Símbolos, +Pontuação, A-Z, a-z, 0-9

// Unicode
// Quantidade de Bytes Variável - Expansível
// Suporta mais de 1 Milhão caracteres
// Atualmente tem mais de 100.000 caracteres atribuidos

// https://unicode-table.com/pt/

const texto = 'a c d'
console.log(texto.match(/\u02AC|\u0BF5/g))
```

# 4. Conjuntos

## 4.1. Trabalhando com Conjuntos

### Listagem 18 - Conjuntos

conjuntos/Conjuntos.js

```
const texto = '1,2,3,4,5,6,a.b c!d?e[f'  
  
// para definir uma classe (ou conjunto) de caracteres usa []  
const regexPares = /[02468]/g  
console.log(texto.match(regexPares))  
  
const texto2 = 'João não vai passear na moto.'  
const regexComESemAcento = /n[aã]/g  
console.log(texto2.match(regexComESemAcento))
```

## 4.2. Trabalhando com Intervalos

### Listagem 19 - Intervalos

conjuntos/Intervalos.js

```
const texto = '1,2,3,4,5,6,a.b c!d?e[f'  
  
console.log(texto.match(/[a-z]/g))  
console.log(texto.match(/[b-d]/g))  
console.log(texto.match(/[2-4]/g))  
console.log(texto.match(/[A-Z1-3]/gi))
```

## 4.3. Conjuntos e Meta-Caracteres

## Listagem 20 - Conjuntos com meta-caracteres

conjuntos/ConjuntosComMetacaracteres.js

```
const texto = '.+$*?-'

console.log(texto.match(/[.+*$/]./g)) // não precisa de escape dentro do conjunto
console.log(texto.match(/[$-?]/g)) // isso é um intervalo (range)

// NÃO é um intervalo (range)...
console.log(texto.match(/[$\-?]/g))
console.log(texto.match(/[-?]/g))

// pode precisar de escape dentro do conjunto: - [ ] ^
```

## 4.4. Alguns Cuidados com Intervalos

### Listagem 21 - Alguns cuidados com intervalos

conjuntos/IntervalosCuidados.js

```
const texto = 'ABC [abc] a-c 1234'

console.log(texto.match(/[a-c]/g))
console.log(texto.match(/a-c/g)) // não define um range

console.log(texto.match(/[A-z]/g)) // intervalos usam a ordem da tabela UNICODE

// tem que respeitar a ordem da tabela UNICODE
// console.log(texto.match(/[a-Z]/g))
// console.log(texto.match(/[4-1]/g))
```

## 4.5. Usando Shorthands

## Listagem 22 - Shorthands

conjuntos/Shorthands.js

```
const texto = '1,2,3,4,5,6,a.b c!d?e\r - f_g'

console.log(texto.match(/\d/g)) // números [0-9]
console.log(texto.match(/\D/g)) // não números [^0-9]

console.log(texto.match(/\w/g)) // caracteres [a-zA-Z0-9_]
console.log(texto.match(/\W/g)) // não caracteres [^a-zA-Z0-9_]

console.log(texto.match(/\s/g)) // espaço [ \t\n\r\f]
console.log(texto.match(/\S/g)) // não espaço [^ \t\n\r\f]

// \b \B
```

## 4.6. Conjuntos Negados

### Listagem 23 - Conjuntos negados

conjuntos/ConjuntosNegados.js

```
const texto = '1,2,3,a.b c!d?e[f'

console.log(texto.match(/\D/g))
console.log(texto.match(/[^0-9]/g))
console.log(texto.match(/[^d!?\s,\.]/g))

const texto2 = '1: !"#$%&\(')*+,.-./ 2: :;=>?@'

console.log(texto2.match(/[^-:@\s]/g))
```

## 4.7. Selezionando Intervalos Unicode

### Listagem 24 - Conjunto com caracteres Unicode

conjuntos/ConjuntoUnicode.js

```
const texto = 'áéíóú àèìòù âêîôû ç ãõ ü'
console.log(texto.match(/[À-ü]/g))
```

# 5. Quantificadores

## 5.1. Conhecendo os Quantificadores

### Metacaracteres - Quantificadores

Metacaractere	Nome	Significado
?	Opcional	Zero ou um
*	Asterisco	Zero ou mais
+	Mais	Um ou mais
{n, m}	Chaves	De n até m

Figura 8. Lista de Quantificadores

### Exemplos de Quantificadores

Quantificador	Significado
{2,3}	De dois a três
{3,}	Três ou mais
{0,4}	Até quatro
{2}	Exatamente dois
{0,1}	Zero ou Um (semelhante ao ?)
{0,}	Zero ou mais (semelhante ao *)
{1,}	Um ou mais (semelhante ao +)

Figura 9. Exemplos de Quantificadores (Chave)

## 5.2. Quantificador: ? (Zero-Um)

### Listagem 25 - ? (Zero-Um)

quantificadores/ZeroUm.js

```
const texto1 = 'De longe eu avistei o fogo e uma pessoa gritando: FOGO000000!'
const texto2 = 'There is a big fog in NYC'

// ? -> zero ou um (opcional)
const regex = /fogo?/gi
console.log(texto1.match(regex))
console.log(texto2.match(regex))
```

## 5.3. Quantificador: + (Um-Mais)

### Listagem 26 - + (Um-Mais)

quantificadores/UmMais.js

```
const texto1 = 'De longe eu avistei o fogo e uma pessoa gritando: FOGOOOOOO!'
const texto2 = 'There is a big fog in NYC'

// + -> um ou mais
const regex = /fogo+/gi
console.log(texto1.match(regex))
console.log(texto2.match(regex))

const texto3 = 'Os números: 0123456789.'
console.log(texto3.match(/[0-9]/g))
console.log(texto3.match(/[0-9]+/g))
```

## 5.4. Quantificador: \* (Zero-Mais)

### Listagem 27 - \* (Zero-Mais)

quantificadores/ZeroMais.js

```
const texto1 = 'De longe eu avistei o fogo e uma pessoa gritando: FOGOOOOOO!'
const texto2 = 'There is a big fog in NYC'

// * -> zero ou mais
const regex = /fogo*/gi
console.log(texto1.match(regex))
console.log(texto2.match(regex))
```

## 5.5. Quantificador: {n, m}

## Listagem 28 - Quantificadores com chaves

quantificadores/UsandoChaves.js

```
const texto = 'O João recebeu 120 milhões apostando 6 9 21 23 45 46.'  
  
// para definir uma quantificador usa {}  
console.log(texto.match(/\d{1,2}/g))  
console.log(texto.match(/[0-9]{2}/g))  
console.log(texto.match(/\d{1,}/g))  
  
console.log(texto.match(/\w{7}/g))  
console.log(texto.match([\wõã]{7,}/g))  
  
// no futuro...  
console.log(texto.match(/\b\d{1,2}\b/g))  
console.log(texto.match(/\b[\wõ]\{7\}\b/g))
```

## 5.6. Guloso vs Não Guloso

### Listagem 29 - Guloso vs Não Guloso

quantificadores/NaoGuloso.js

```
const texto = '<div>Conteudo 01</div><div>Conteudo 02</div>'  
  
// quantificadores SÃO gulosos (greedy)...  
console.log(texto.match(<div>.+</div>/g))  
console.log(texto.match(<div>.*</div>/g))  
console.log(texto.match(<div>.{0,100}</div>/g))  
  
// quantificadores NÃO gulosos (lazy)...  
console.log(texto.match(<div>.+?</div>/g))  
console.log(texto.match(<div>.*?</div>/g))  
console.log(texto.match(<div>.{0,100}?</div>/g))
```

## 5.7. Apresentando Desafios

## Listagem 30 - Os desafios

quantificadores/Desafios.txt

```
// DesafioCPF.js
'CPF dos aprovados:
- 600.567.890-12
- 765.998.345-23 ...'

// DesafioTelefone.js
'Lista telefônica:
- (11) 98756-1212
- 98765-4321 ...'

// DesafioEmail.js
'Os e-mails dos convidados são:
- fulano@cod3r.com.br
- xico@gmail.com ...'
```

## 5.8. Desafio: Selecionando CPF

### Listagem 31 - Desafio CPF

quantificadores/DesafioCPF.js

```
const texto =
'CPF dos aprovados:
- 600.567.890-12
- 765.998.345-23
- 454.674.333-21
- 678.987.123-87
- 789.112.543-00
'

console.log(texto.match(/\d{3}.\d{3}.\d{3}-\d{2}/g))
```

## 5.9. Desafio: Selecionando Telefone

## Listagem 32 - Desafio Telefone

quantificadores/DesafioTelefone.js

```
const texto = `Lista telefônica:  
- (11) 98756-1212  
- 98765-4321  
- (85) 99988-7766  
- (21)3261-8899`  
  
console.log(texto.match(/\((?!\d{0,2})\)?\s?\d{4,5}-\d{4}/g))
```

## 5.10. Desafio: Selecionando E-mail

### Listagem 33 - Desafio E-mail

quantificadores/DesafioEmail.js

```
const texto = `Os e-mails dos convidados são:  
- fulano@cod3r.com.br  
- xico@gmail.com  
- joao@empresa.info.br  
- maria_silva@registro.br  
- rafa.sampaio@yahoo.com`  
  
console.log(texto.match(/[a-zA-Z0-9_]+@[a-zA-Z0-9_]+\.[a-zA-Z0-9_]{2,4}/g))  
console.log(texto.match(/\w+@\w+\.\w{2,4}/g))  
console.log(texto.match(/[\w.]+@\w+\.\w{2,4}/g))  
console.log(texto.match(/[\w.]+@\w+\.\w{2,4}\.?\w{0,2}/g))  
  
// no futuro...  
console.log(texto.match(/[\w.]+@\w+\.\w{2,4}(\.\w{2})?/g))
```

# 6. Grupos

## 6.1. Usando Grupos

### Listagem 34 - Grupos

grupos/Grupos.js

```
const texto1 = 'O José Simão é muito engraçado... hehehehehe'
console.log(texto1.match(/(he)+/g))

const texto2 = 'http://www.site.info www.escola.ninja.br google.com.ag'
console.log(texto2.match(/(http:\/\/)?(www\.)?\w+\.\w{2,}(\.\w{2})?/g))
```

## 6.2. Grupos & Retrovisores

### Listagem 35 - Retrovisor

grupos/Retrovisor.js

```
const texto1 = '<b>Destaque</b><strong>Forte</strong><div>Conteúdo</div>'
console.log(texto1.match(/<(\w+)>.*<\/\1>/g))

const texto2 = 'Lentamente é mente muito lenta.'
console.log(texto2.match(/(lenta)(mente).*\2.*\1/gi))
console.log(texto2.match(/(?:lenta)(mente).*\1/gi)) // ?: não guarda

console.log(texto2.match(/(lenta)(mente)/gi))
console.log(texto2.match(/(lenta)(mente)?/gi))
console.log(texto2.replace(/(lenta)(mente)/gi, '$2'))
```

## 6.3. Retrovisores: Só por Curiosidade

### Listagem 36 - Retrovisor

grupos/Retrovisor.js

```
const texto1 = '<b>Destaque</b><strong>Forte</strong><div>Conteúdo</div>'  
console.log(texto1.match(/<(\w+)>.*<\/\1>/g))  
  
const texto2 = 'Lentamente é mente muito lenta.'  
console.log(texto2.match(/(lenta)(mente).*\2.*\1./gi))  
console.log(texto2.match(/(?:lenta)(mente).*\1/gi)) // ?: não guarda  
  
console.log(texto2.match(/(lenta)(mente)/gi))  
console.log(texto2.match(/(lenta)(mente)?/gi))  
console.log(texto2.replace(/(lenta)(mente)/gi, '$2'))  
  
const texto3 = 'abcdefghijkl'  
console.log(texto3.match(/(a)(b)(c)(d)(e)(f)(g)(h)(i)(j)(k)(l)\12/g))
```

## 6.4. Grupos Aninhados

### Listagem 37 - Grupos aninhados

grupos/GruposAninhados.js

```
const texto = 'supermercado hipermercado minimercado mercado'  
  
console.log(texto.match(/(super|hiper|mini)?mercado/g))  
console.log(texto.match(/((hi|su)per|mini)?mercado/g))
```

## 6.5. Alguns Cuidados com Grupos

### Listagem 38 - Alguns cuidados com grupos

grupos/GruposAlgunsCuidados.js

```
const texto = 'Pedrinho (filho do Pedro Silva) é doutor do ABCabc!'  
  
console.log(texto.match(/[(abc)]/gi)) // não é um grupo  
console.log(texto.match(/([abc])/gi))  
console.log(texto.match(/(abc)+/gi))
```

## 6.6. Grupos Especiais #01

### Listagem 39 - Grupos especiais #01

grupos/GruposEspeciais1.js

```
const texto = 'João é calmo, mas no transito fica nervoso.'  
  
console.log(texto.match(/[\wÀ-ú]+/gi))  
  
// Positive lookahead  
console.log(texto.match(/[\wÀ-ú]+(?=, )/gi))  
console.log(texto.match(/[\wÀ-ú]+(?=\.\.)/gi))  
console.log(texto.match(/[\wÀ-ú]+(?=, mas)/gi))  
  
// Negative lookahead  
console.log(texto.match(/[\wÀ-ú]+\\b(?! ,)/gi))  
console.log(texto.match(/[\wÀ-ú]+[\\s|\\.](?! ,)/gi))
```

## 6.7. Grupos Especiais #02

### Listagem 40 - Grupos especiais #02

grupos/GruposEspeciais2.rb

```
texto = 'supermercado superação hiperMERCADO Mercado'  
  
puts texto.scan(/(?:super)[\wÀ-ú]+/i).join(' ')  
  
# Positive Lookbehind  
puts texto.scan(/(?<=super)[\wÀ-ú]+/i).join(' ')  
  
# Negative Lookbehind  
puts texto.scan(/(?<!super)mercado/i)
```

# 7. Bordas

## 7.1. Usando Bordas

### Listagem 41 - *Bordas*

bordas/Bordas.js

```
const texto = 'Romário era um excelente jogador\n, mas hoje é um político questionador'

console.log(texto.match(/r/gi))
console.log(texto.match(/^r/gi)) // ^ inicio da linha/string
console.log(texto.match(/r$/gi)) // $ fim da linha/string

console.log(texto.match(/^r.*r$/gi)) // problema do dotall
```

## 7.2. Implementando Dotall em JS

### Listagem 42 - *Dotall em JS*

bordas/Dotall.js

```
const texto = 'Romário era um excelente jogador\n, mas hoje é um político questionador'

console.log(texto.match(/^r.*r$/gi)) // problema do dotall
console.log(texto.match(/^r[\s\S]*r$/gi))
```

## 7.3. Usando a Flag Multiline

### Listagem 43 - Flag Multiline

bordas/Multiline.js

```
const texto = `  
Leo é muito legal  
Emanuel foi jogar em Sergipe  
Bianca é casada com Habib  
  
console.log(texto.match(/\n/g))  
console.log(texto.match(/^(\w).+\1$/gi))  
console.log(texto.match(/^(\w).+\1$/gim))
```

## 7.4. Bordas de Palavras

### Listagem 44 - Borda de palavra

bordas/BordaPalavra.js

```
const texto1 = 'dia diatônico diafragma média wikipedia bom_dia melodia radial'  
  
console.log(texto1.match(/\bdia\w+/gi))  
console.log(texto1.match(/\w+dia\b/gi))  
console.log(texto1.match(/\w+dia\w+/gi))  
console.log(texto1.match(/\bdia\b/gi))  
  
// borda é não \w, que é [^A-Za-z0-9_]... temos problemas com acentos!  
const texto2 = 'dia diatônico diafragma, média wikipedia bom-dia melodia radial'  
console.log(texto2.match(/\bdia\b/gi))  
console.log(texto2.match(/(\S*)?dia(\S*)?/gi)) // a vírgula entra!  
console.log(texto2.match(/([\wÀ-ú-]*)?dia([\wÀ-ú-]*)?/gi))
```

## 8. Receitas (Exercícios)

### 8.1. Aplicando Syntax Highlight #01

**Listagem 45 - Responsável por ler e escrever arquivo**

`receitas/files.js`

```
const fs = require('fs')

const read = nomeArquivo =>
  fs.readFileSync(`${__dirname}/originais/${nomeArquivo}`, {encoding: 'utf8'})

const write = (nomeArquivo, conteudo) => {
  const dirname = `${__dirname}/alterados`
  if (!fs.existsSync(dirname)) {
    fs.mkdirSync(dirname)
  }

  fs.writeFileSync(`${dirname}/${nomeArquivo}`, conteudo, {encoding: 'utf8'})
}

module.exports = { read, write }
```

## Listagem 46 - Página com código fonte

receitas/originais/codigoFonte.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Código Fonte</title>
    <style>
        code { font-size: 26px; }
    </style>
</head>
<body>
    <h1>Código Fonte</h1>
    <pre>
        <code>
package cod3r;

/*
 * Imprimir a nota do aluno
 */
public class Nota {

    // Porta de entrada de um programa Java
    public static void main(String[] args) {
        int nota = 7;

        if(nota >= 7) {
            System.out.println("Aprovado");
        } else {
            System.out.println("Reprovado");
        }
    }
}
        </code>
    </pre>
</body>
</html>
```

## 8.2. Aplicando Syntax Highlight #02

## Listagem 47 - Colorir código fonte

receitas/ColorirCodigoFonte.js

```
const aplicarCor = (txt, reg, cor) => txt.replace(reg, `<span style="color: #${cor}"><b>$1</b></span>`)

const files = require('./files')
const texto = files.read('codigoFonte.html')

const codeRegex = /<code>[\s\S]*</code>/i
let codigo = texto.match(codeRegex)[0]

// String...
codigo = aplicarCor(codigo, /(\".*\")/g, 'ce9178')

// palavras reservadas
codigo = aplicarCor(codigo, /\b(package|public|class|static|if|else)\b/g,
'd857cf')

// tipos...
codigo = aplicarCor(codigo, /\b(void|int)\b/g, '1385e2')

// comentários de multiplas linhas...
codigo = aplicarCor(codigo, /(\/\*[ \s\S]*\*\//g, '267703')

// comentários de uma linha...
codigo = aplicarCor(codigo, /(\/\/.*?\n)/g, '267703')

const conteudoFinal = texto.replace(codeRegex, codigo)
files.write('codigoFonte.html', conteudoFinal)
```

## 8.3. Selecionando Telefones

### Listagem 48 - Receita para selecionar telefone

receitas/Telefones.js

```
const texto = `  
Lista telefônica:  
- (21) 12345-6789  
- (11) 62300-2234  
- 5678-7771  
- (85)3333-7890  
- (1) 4321-1234  
  
console.log(texto.match(/((\d{2})\s?)?\d{4,5}-\d{4}/g))
```

## 8.4. Selecionando Intervalos Numéricos

### Listagem 49 - Receita para selecionar intervalo numérico

receitas/IntervaloNumerico.js

```
const texto = '0 1 10 192 199 201 249 255 256 312 1010 1512'  
  
// números entre 0-255  
console.log(texto.match(/\b(\d{1,2}|1\d{2}|2[0-4]\d|25[0-5])\b/g))
```

## 8.5. Selecionando Endereços IPv4

## Listagem 50 - Receita para selecionar IPv4

receitas/Ipv4.js

```
const texto = `  
Inválidos:  
192.268.0.1  
1.333.1.1  
192.168.0.256  
256.256.256.256  
  
Válidos:  
192.168.0.1  
127.0.0.1  
10.0.0.255  
10.11.12.0  
255.255.255.255  
0.0.0.0  
`  
  
const n = `(\d{1,2}|\d{2}|2[0-4]\d|25[0-5])`  
const ipv4 = RegExp(`\b${n}\.${n}\.${n}\.${n}\b`, 'g')  
console.log(texto.match(ipv4))
```

## 8.6. Validação de Senha

### Listagem 51 - Receita para validar senha

receitas/Senha.js

```
const texto = `  
123456  
cod3r  
QUASE123!  
#0pA1  
#essaSenhaEGrande1234  
  
#0pA1?  
Foi123!  
`  
  
console.log(texto.match(/^.{6,20}$/gm))  
console.log(texto.match(/^([A-Z]).{6,20}$/gm))  
console.log(texto.match(/^([A-Z])([a-zA-Z])([0-9])([@#$%^&*]).{6,20}$/gm))
```

## 8.7. Selecionando E-mail

### Listagem 52 - \_Receita para selecionar e-mail

receitas/Email.js

```
const texto = `Os e-mails dos convidados são:  
- fulano@cod3r.com.br  
- xico@gmail.com  
- joao@empresa.info.br  
- maria_silva@registro.br  
- rafa.sampaio@yahoo.com  
- fulano+de+tal@escola.ninja.br`  
  
console.log(texto.match(/\S+@\w+\.\w{2,6}(\.\w{2})?/g))
```

# **9. Conclusão**

## **9.1. Obrigado e Até Breve**

Obrigado e até Breve!

# Appendix A: Tabela de Códigos

- Listagem 1 - *Flags*
- Listagem 2 - *Usando JS*
- Listagem 3 - *Usando Ruby*
- Listagem 4 - *Usando Python*
- Listagem 5 - *Usando Go*
- Listagem 6 - *Usando Java*
- Listagem 7 - *Olá Regex*
- Listagem 8 - *Alguns Cuidados*
- Listagem 9 - *Caractere Simples*
- Listagem 10 - *Meta-caracteres*
- Listagem 11 - *Meta-caractere Ponto*
- Listagem 12 - *Desafio Lista de Arquivos*
- Listagem 13 - *Caracteres Brancos*
- Listagem 14 - *Desafio dos três espaços*
- Listagem 15 - *Pipe (Ou)*
- Listagem 16 - *Problema do Ponto*
- Listagem 17 - *Caracteres Unicode*
- Listagem 18 - *Conjuntos*
- Listagem 19 - *Intervalos*
- Listagem 20 - *Conjuntos com meta-caracteres*
- Listagem 21 - *Alguns cuidados com intervalos*
- Listagem 22 - *Shorthands*
- Listagem 23 - *Conjuntos negados*
- Listagem 24 - *Conjunto com caracteres Unicode*
- [ex-05\_010\_quantificadores]
- Listagem 25 - *? (Zero-Um)*
- Listagem 26 - *+ (Um-Mais)*
- Listagem 27 - *\* (Zero-Mais)*
- Listagem 28 - *Quantificadores com chaves*
- Listagem 29 - *Guloso vs Não Guloso*
- Listagem 30 - *Os desafios*
- Listagem 31 - *Desafio CPF*

- Listagem 32 - *Desafio Telefone*
- Listagem 33 - *Desafio E-mail*
- Listagem 34 - *Grupos*
- Listagem 35 - *Retrovisor*
- Listagem 36 - *Retrovisor*
- Listagem 37 - *Grupos aninhados*
- Listagem 38 - *Alguns cuidados com grupos*
- Listagem 39 - *Grupos especiais #01*
- Listagem 40 - *Grupos especiais #02*
- Listagem 41 - *Bordas*
- Listagem 42 - *Dotall em JS*
- Listagem 43 - *Flag Multiline*
- Listagem 44 - *Borda de palavra*
- Listagem 45 - *Responsável por ler e escrever arquivo*
- Listagem 46 - *Página com código fonte*
- Listagem 47 - *Colorir código fonte*
- Listagem 48 - *Receita para selecionar telefone*
- Listagem 49 - *Receita para selecionar intervalo numérico*
- Listagem 50 - *Receita para selecionar IPv4*
- Listagem 51 - *Receita para validar senha*
- Listagem 52 - *\_Receita para selecionar e-mail*

# Glossário